Search Computing Meets Data Extraction∗

Alessandro Bozzon, Marco Brambilla, Chiara Pasini, Luca Tettamanti, Salvatore Vadacca
Politecnico di Milano, Dipartimento di Elettronica ed Informazione, Via Ponzio 34/5, 20133 Milano, Italy
firstname.lastname@elet.polimi.it

Tim Furche, Giorgio Orsi
Oxford University, Department of Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD
firstname.lastname@comlab.ox.ac.uk

ABSTRACT
Thanks to the Web, access to an increasing wealth and variety of information has become near instantaneous. To make informed decisions, however, we often need to access data from many different sources and integrate different types of information. Manually collecting data from scores of web sites and combining that data remains a daunting task.

The ERC projects SeCo (Search Computing) and DIADEM (Domain-centric Intelligent Automated Data Extraction Methodology) address two aspects of this problem. SeCo supports complex search processes drawing on data from multiple domains with a user interface capable of refining and exploring the search results. DIADEM aims to automatically extract structured data from any website of a given domain.

In this paper, we outline a first approach for integrating SeCo and DIADEM. In particular, we discuss how to use the DIADEM methodology to turn nearly any website from a given domain into a SeCo search service. We describe how such services can be registered and exploited by the SeCo framework in combination with services from other domains (and possibly developed with other methodologies).

1. INTRODUCTION
Recent years witnessed a paradigmatic shift in the way people deal with information. While the Web provided cheap and ubiquitous access to an increasing wealth and variety of information, to allow people making informed decisions there is often the need for complex and articulated information retrieval tasks, which typically involve the access to information coming from many different sources. Manually collecting data from scores of Web sites and combining that data remains a daunting task: the query “retrieve jobs as Java Developer in Silicon Valley, nearby affordable fully-furnished flats, and close good schools” is, unfortunately, still not addressed by current search engines, and the “data integration” task is performed by the user.

The Search Computing project (SeCo) [?, ?] aims at building concepts, algorithms, tools and technologies to support complex Web queries, through a new paradigm based on combining data extraction from distinct sources and data integration by means of specialized integration engines.

Web data is typically published in two ways: as structured (and possibly linked) data accessible trough Web APIs (e.g. SPARQL, YQL, etc.), and as unstructured resources (i.e. Web pages), possibly accessible only through user-mediated activities such as form filling or link navigation. Unstructured data is typically accessible to general-purpose search engines, which exploits tradition information retrieval techniques. To enable the consumption of such data by automated processes, data accessible to humans through existing Web interfaces needs to be transformed into structured information: therefore, there is the need for information extraction tool (e.g. screen scrapers); unfortunately, the interactive nature of modern web interfaces poses a big challenge, as the dynamic nature of these user interfaces, driven by client and server-side scripting, creates challenges for automated processes to access this information.

The DIADEM (Domain-centric Intelligent Automated Data Extraction Methodology) project aims at developing domain-specific data extraction systems that take as input a URL of a Web site in a particular application domain, automatically explore the Web site, and deliver as output a structured data set containing all the relevant information present on that site. Toward this end, the OXPath language has been proposed: OXPath is a superset of XPath 1.0. Beyond XPath, OXPath is able (1) to fill web forms and trigger DOM events, (2) to access dynamically computed CSS attributes, (3) to navigate between visible form fields, and (4) to mark relevant information for extraction. This way, OXPath expressions can closely simulate the human interaction relevant for navigation rather than rely exclusively on the HTML structure.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXX-XX-X/XX/XX ...$10.00.
In this paper we briefly outline an integrated approach to Web data search based on the work of the DIADEM and the SeCo project by demonstrating how the two approaches nicely complement each other, one being devoted to information extraction from the Web and the other to integration of such information.

The paper is organized as follows: Section 2 describes the search computing approach to information integration, Section 3 presents the OXPath language as a tool for information extraction, Section 4 discusses the integration issues, and Section 5 concludes.

2. WEB DATA INTEGRATION WITH SEARCH COMPUTING

Figure 1 shows an overview of the Search Computing framework, which comprises several sub-frameworks. The service description framework (SDF) provides the scaffolding for wrapping and registering data sources in service marts, describing the information sources at different levels of abstraction. The user framework provides functionality and storage for registering users, with different roles and capabilities. The query framework supports the management and storage of queries as first class citizens: a query can be executed, saved, modified, and published for other users to see. The service invocation framework masks the technical issues involved in the interaction with the service mart, e.g., the Web service protocol and data caching issues. The core of the framework aims at executing multi-domain queries. The query manager takes care of splitting the query into subqueries (e.g., “Which jobs as Java developer are available in the Silicon Valley?”, “Where are affordable, nearby flats?”, “Where are good schools?”) and bounding them to the respective relevant data sources registered in the service mart repository; starting from this mapping, the query planner produces an optimized query execution plan, which dictates the sequence of steps for executing the query. Finally, the execution engine actually executes the query plan, by submitting the service calls to designated services through the service invocation framework, building the query results by combining the outputs produced by service calls, computing the global ranking of query results, and producing the query result outputs in an order that reflects their global relevance.

3. AUTOMATIC WEB INFORMATION EXTRACTION IN DIADEM

For many people, the web is nowadays one of the primary sources of information. As a consequence, almost every business must publish and maintain a website to stay operational. Having a web visibility is especially important for those businesses operating in very dynamic markets such as job advertising, retail products and real-estate. As an example, through the web a small real-estate agency can be as visible as a large real-estate group. However, these companies usually spend considerable effort to be ranked by search engines such as Google and Bing as well as to be integrated into major aggregators and portals.

With time, businesses have become more and more dependent on dominant aggregators and search engines where it is hard to distinguish oneself by reputation, service, or novel features rather than price. Google, on the other hand, is able to pick up on reputation of agencies to some extent, but does very poorly on property searches. This observation holds nowadays for many product domains, with price comparison systems becoming dominant.

One of the goals of DIADEM is to address this dependency on few, dominant aggregators by providing a methodology to automatically extract published web data into a structured format (e.g., RDF files or web services) whose schema is represented in a machine readable format (e.g., an RDF ontology or a WSDL document).

Search engines can be then adapted for object search, i.e., finding the property best fitting specific criteria (such as price range and location). In contrast to aggregators, object search engines should be able to pick up any properly structured data published on the web, just as search engines pick up any properly published HTML page.

Previous approaches to fully-automated data extraction addressed the problem by investigating general techniques that can be applied to any web site. The results have been poor since the variety of web-site modelling paradigms as well as the technologies in the web is cumbersome. On the contrary, DIADEM is based on a fundamental observation: if we combine domain with phenomenological (object and search) knowledge in an ontology of a domain’s web objects, we can automatically derive an extraction program for nearly any web page in the domain. The resulting program produces high precision data, as we use domain knowledge to improve recognition and alignment and to verify the extraction program based on ontological constraints.

Diadem’s prototype architecture is sketched in Figure 1. The input of the Diadem approach are the URLs of the web sites we want to analyze. Every web page is processed in a sequential fashion. First we extract the page model from a live rendering of the web page (see Figure 2). This model logically represents the DOM tree of the page along...
with information on the visual rendering (e.g., CSS boxes), and linguistic annotations. The information provided by the browser model is mainly domain-independent (e.g., DOM structure and CSS boxes) while some of the linguistic annotations are generated by domain-specific gazetteers and rules. In the next step, we do an initial classification of web blocks, such as navigation links and advertisements areas, to separate general structures and interesting data from noise structures and to provide additional clues to form analysis. In the third step we identify and classify any objects of the domain that may occur on the page. This is done before the form analysis in stage four, as we use the information from the object analysis together with the block classification to decide if navigation links on the page may lead to further data. If the form analysis can identify a form belonging to the target domain, we proceed to fill that form (possibly multiple times). The obtained result-pages are then analysed in order to identify records containing interesting data. The analysis starts by locating mandatory attributes of the records that we expect to find on a web page of a given domain; then, we proceed to the segmentation of the page into records through domain-independent heuristics. The identified records are then validated using a result-page model.

Once a site is fully explored, all collected models are passed to the OXPath generator that uses simple heuristics to create a generalised OXPath expression that to be executed with the OXPath prototype for large scale extraction (see Section ??). This analysis only needs to be repeated if the analyzed site has changed significantly, otherwise the resulting OXPath expression can be used for repeated extraction from the web site.

4. TOWARD MULTI-DOMAIN, AUTOMATED WEB DATA CONSUMPTION

Our approach for the integration of structured and unstructured Web data sources is based on a service-oriented vision of the resources. The source integration operates at three levels: wrapping, registration, and invocation.

Service wrapping consists in implementing appropriate wrapping components that take care of invoking the services and manipulating the input and output so as to be consistent with the formats expected by the integration platform. The SeCo platform natively supports generic web services, relational databases, YQL services, SPARQL endpoints, etc. However, the system is open to support additional data source types, so we can provide appropriate integration of OXPath data sources. The wrapper of OXPath sources will consist of a component configured by URL of the starting Web page plus a list of OXPath queries. The wrapper will provide in output the extracted data, formatted according to the SeCo conventions. Adding support for OXPath data sources concretely requires the implementation of a new service invoker (that shares the management of HTTP connections with other invokers, e.g. the YQL invoker, and hence inherits from the same abstract HTTP invoker); it wraps the OXPath engine to automatically parse the output of an OXPath query and adapts it to the expected output schema, as defined in the OXPath source description.

Service description in SeCo is based on the registration of services within the Service Description Framework model, which describes services at three levels of abstraction: Service Marts (abstractions of several Web services dealing with the same conceptual objects available on the Web such as “flights”, “hotels”, and “restaurants”), Access Patterns (a specific signature of the Service Mart with the characterization of each attribute as input, output, and/or ranking), and service interfaces (a description of the invocation interface of an actual source service) - leading from the conceptual representation of Web objects to the implementation of search services. Data produced by unstructured data sources and extracted by OXPath expressions can be easily integrated, because their registration can take place similarly to standard ones.

Service execution is performed by an engine, which exploits a custom view of the Service Description Framework. The execution engine consists of a runtime (a Panta Rhei interpreter able to translate an execution plan in a coordinated sequence of service invocations) and a set of service invokers. Low-level service invokers are implemented (one for each data source type, including the one for OXPath
Figure 5: UML class diagram of the SeCo invokers

sources) and follow the chain of responsibility pattern (see Figure 2). An high-level caching invoker wraps the sequence of low-level invokers to read results from cache.

5. CONCLUSIONS – 0.5 PAGE